The Parallel Implementation of Configuration-Selecting Multireference Configuration Interaction Method

P. STAMPFUSS, W. WENZEL, H. KEITER

Institut für Physik, Universität Dortmund, D-44221 Dortmund, Germany

Received 10 February 1999; accepted 8 June 1999

ABSTRACT: We report on a scalable implementation of the configuration-selecting multireference configuration interaction method for massively parallel architectures with distributed memory. Based on a residue driven evaluation of the matrix elements, this approach allows the routine treatment of Hilbert spaces of well over 10^9 determinants, as well as the selective treatment of triple and quadruple excitations with respect to the reference space. We demonstrate the scalability of the method for up to 128 nodes on the IBM-SP2 and for up to 256 nodes on the Cray-T3E. We elaborate on the specific adaptation of the transition residue-based matrix element evaluation scheme that ensures the scalability and load balancing of the method. © 1999 John Wiley & Sons, Inc. J Comput Chem 20: 1559–1570, 1999

Keywords: configuration-selecting multireference configuration interaction method (MRCI); massively parallel computation; transition residue-based matrix element evaluation; scalable implementation; benchmark methods

Introduction

For many years the multireference configuration interaction method (MRCI)[1–3] has been one of the benchmark tools for highly accurate calculations of the electronic structure of atoms and molecules. Ever since the development of the direct CI algorithm,[1] which obviates the explicit storage of the CI matrix, highly efficient implementations[4] have been used for a wide variety of molecules. The generic lack of extensivity of the MRCI method has at least been partially addressed with a number of *a posteriori*[5,6] corrections and through direct modification of the CI energy functional.[7–12]

Due to its high computational cost, however, many applications of MRCI remain constrained to relatively small systems. For this reason the con-

Correspondence to: W. Wenzel; e-mail: wenzel@wap.physik.uni-dortmund.de

Contract/grant sponsor: DFG; contract/grant number: KEI-164/11-2

CCC 0192-8651 / 99 / 141559-12

figuration-selective version of the MRCI method (MRD-CI), which was introduced by Buenker and Peyerimhoff,[13–15] has arguably become one of its most widely used versions. In this variant only the most important configurations of the interacting space of a given set of primary configurations are chosen for the variational wave function, while the energy contributions of the remaining configurations are estimated on the basis of second-order Rayleigh–Schrödinger perturbation theory.[16,17] A configuration is selected for the variational wave function if its perturbative energy contribution or coefficient is above a given threshold λ and the total energy (the sum of the variational and the perturbative contributions) is extrapolated to the limit $\lambda \to 0$. While this extrapolation is known to fail in isolated instances, it gives a remarkably good resolution of relative energies across the potential energy surface (PES) in the overwhelming majority of applications. Because the variationally treated subspace of the problem consists of only a fraction of the overall Hilbert space, the determination of eigenstates in the truncated space requires far less computational effort. Indeed, for typical applications the overwhelming majority of the computational effort is concentrated in the expansion loop in which the energy contribution of candidate configurations is computed.

Even within this approximation, the cost of MRCI calculations remains rather high. The development of efficient configuration-selecting CI codes[17–22] is inherently complicated by the sparseness and the lack of structure of the selected state vector. In order to further extend the applicability of the method, it is thus desirable to employ the most powerful computational architectures available for such calculations. Here we report on the progress of the first massively parallel, residue-driven implementation of the MRD-CI method for distributed memory architectures. While efforts to parallelize standard MR-SDCI (all single and double excitations) on distributed memory architectures face significant difficulties rooted in the need to distribute the CI vectors over many nodes,[23–26] a parallel implementation of MRD-CI can capitalize on the compactness of its state representation. In our implementation the difficulty of the construction of the subset of nonzero matrix elements is overcome by the use of a residue-based representation of the matrix elements that was originally developed for the distributed memory implementation of MR-SDCI.[26] This approach allows us to efficiently evaluate the matrix elements in the expansion loop, as well as during the variational improvement of the coefficients of the selected vectors.

Going somewhat beyond the standard MRD-CI, our implementation was specifically optimized to estimate the importance of triple and quadruple excitations of the reference configuration. The energy arising from such configurations yields the overwhelming contribution to the energy difference between FCI and MR-SDCI and is thus of paramount importance for the development of approximately extensive versions of the MRCI method.[7–10] Because the number of higher than doubly excited configurations rises so quickly with the system size, FCI and CI-SDTQ calculations are prohibitively expensive for all but the smallest systems. Configuration selecting CI thus provides a particularly effective, maybe the only viable, compromise between computational efficiency and accuracy. Although we do not advocate the generic inclusion of triple and quadruple (TQ) excitations in MR-DCI calculation, we hope that our implementation can be useful to evaluate the performance of approximate extensivity corrections, most importantly for incomplete reference spaces.

In this manuscript we focus on the details of the implementation of the method and provide timings for benchmark applications that demonstrate the scalability of the method for up to 128 nodes of an IBM-SP2 and up to 256 nodes of a Cray-T3E for Hilbert spaces of dimensions up to 5×10^9 of which up to 5×10^6 elements were selected for the variational wave function. The determinant-based code we report here was developed in an object oriented implementation using C++ as the implementation language and is available with the Dortmund quantum chemistry package.

Methodology

In the following we describe the key ingredients for the residue-based parallel implementation of the configuration selecting MR-CI method. We begin with a description of the orbital partitioning scheme that allows a flexible treatment of the triple and quadruple excitations with regard to the active space. We then describe the principle of the residue-based matrix-element evaluation that is at the heart of our algorithm, followed by a more detailed treatment of the key difficulties in the two

computationally expensive phases of the method: the expansion step and the iteration step. Next we present the results of benchmarks of the method for O_2, NO_2, and benzene as a function of the number of nodes used.

PARTITIONING OF ORBITAL SPACE

In order to facilitate the explicit treatment of triple and quadruple excitations from a given reference set, we partition the orbital space into five segments according to a partition scheme that proved promising in an earlier investigation.[27] We have the i(nactive) subspace of orbitals that must be doubly occupied in all reference configurations, the a(ctive) space containing all orbitals that appear in at least one reference, the l(ow) space containing a set of orbitals into which triple- and quadruple excitations are allowed, the h(igh) space into which excitations of the type (llhh) (i.e., double occupancy in the low and double occupancy in the high) or (lllh) are allowed, and the d-space containing all orbitals into which only double and single excitations are allowed. In a traditional MRD-CI calculation the orbital sets (l) and (h) are empty. The population of these segments allows the gradual inclusion of TQ excitations in cases where even the enumerative search of such configurations in the expansion loop is prohibitively expensive.

An earlier investigation into the hierarchy of TQ excitations demonstrated that in many cases the natural orbital occupations in the single and double (SD) segment of the configuration space is clearly separated in weight by the natural orbital occupation arising from the TQ segment. As a result, the fraction of the overall natural orbital (NO) basis that contributes to important TQ configurations at some given threshold is much smaller than the fraction of the NO basis that contributes to SD configurations at the same threshold. This observation is illustrated in Figure 1 where we have separated the contributions to the diagonal elements of the single particle density matrix that arise from SD and TQ excitations, respectively. Note that for each orbital the total weight of all TQ excitations lies almost 2 orders of magnitude below the weight arising from SD contributions. This observation suggests that it is worthwhile to restrict the search of the overall TQ space to the most important orbitals of the SD space.

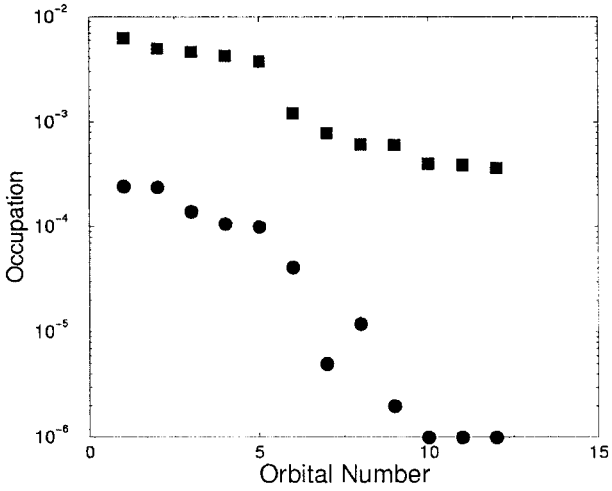

FIGURE 1. Total contributions of individual orbitals (of a_g symmetry) to the diagonal matrix elements of the single-particle density matrix that arise from (■) single and double excitations of the reference configurations or (●) triple and quadruple excitations. The data were obtained in a CI-SDTQ calculation for O_2 in a cc-pVDZ basis at the experimental geometry. The details of the calculation are summarized in the benchmark section.

RESIDUE-BASED EVALUATION OF MATRIX ELEMENTS

In order to compute the matrix elements of the Hamilton operator we exploit an enumeration scheme in which each matrix element between two determinants (or configuration state functions) $|\phi_1\rangle$ and $|\phi_2\rangle$ is associated with the subset of orbitals that occur in both the target and the source determinant. This unique subset of orbitals is called the *transition residue* mediating the matrix element and serves as a sorting criterion to facilitate the matrix element evaluation on distributed memory architectures. For a given many-body state, we consider a tree of all possible transition residues as illustrated in Figure 2. For each such residue we build a list of *residue entries*, which are composed of the orbital pairs (or an orbital for a single-particle residue) that combine with the residue to yield a selected configuration and a pointer to that configuration. Although the number of transition residues is comparatively small, the overall number of residue entries grows rapidly (as $N_{\text{selected}} n_e^2$) with the number of configurations N_{selected} and the number of electrons n_e. In a nonselecting CI calculation it is thus not possible to store the entire data structure at any given time and much effort must be devoted to efficiently evaluate segments of the residue tree, which are discarded after use.[26] For configuration selecting CI, however, the reduction

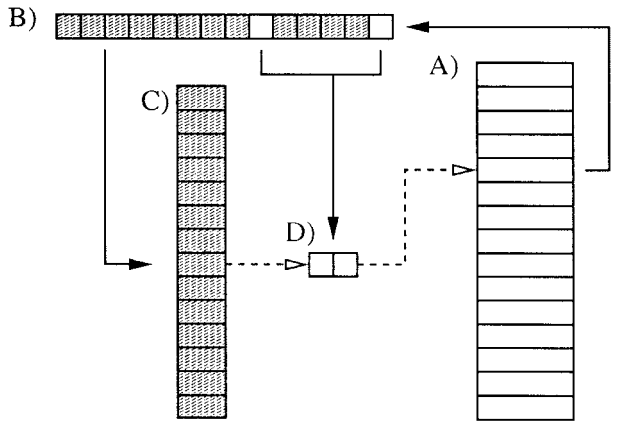

FIGURE 2. Schematic representation of the two-particle residue tree. For each element of the configuration list (A) all possible two-particle residues are constructed. In the configuration illustrated in (B) each box represents one occupied orbital; the shaded region corresponds to the residue, and the two white boxes correspond to the orbital pair. The $(n_e - 2)$ electron residue configuration is looked up in the residue-tree (C), where an element (D) is added that encodes the orbitals that were removed, information regarding the permutation required, and the index of the original configuration in the configuration list. The solid arrows in the figure indicate logical relationships; the dotted arrows indicate pointers incorporated in the data structure. The residue list along with all elements must be rebuilt once after each expansion loop; the effort to do so is proportional to the product of n_e^2 with the number of configurations. The number of matrix elements encoded in a single element of the residue tree is proportional to the *square of the number of entries* of type (D).

in the number of selected configurations combined with the large total memory of modern distributed memory machines allows us to build the residue tree for the selected configurations, provided that only the required sections of the residue tree are stored in the different stages of the computation.

Once the residue tree is available, the evaluation of the matrix elements is very efficient. In the *expansion step* one must evaluate $q_i = \langle \phi_i | HP | \Psi \rangle$, where P projects on the part of the Hilbert space in which only inactive and active and low orbitals are occupied. For SD (single and double excitations only) calculations one needs to consider only residues and orbital pairs that contain no d-type orbitals. For TQ (single and double, triple and quadruple excitations) at most two orbitals of type h and l are allowed in the residue and none in the orbital pair. This portion of the residue tree contains but a fraction of the overall residue tree and can be accommodated on all nodes. For each $|\phi_i\rangle$ we determine the required single- and two-particle residues, which are then searched for in the residue tree. In an SD calculation one can eliminate the search step by constructing the allowed excitations directly from the internal residues. If a match is found the information in the tree enables us to immediately compute *all matrix elements* associated with the given residue. As a result, the overall numerical effort scales linearly with the number of configurations for which matrix elements must be evaluated and the number of nontrivial operations per configuration is proportional to n_e^2. Again, there is a significant difference between configuration-selecting and nonselecting CI calculations. In nonselecting CI the number of nonzero matrix elements scales as $n_e^2 N^2$ for each configuration. In configuration-selecting CI most of the possible matrix elements do not lead to selected configurations; thus, the number of entries per element of the transition residue tree is much smaller. The residue-based implementation avoids the explicit enumeration of matrix elements that do not lead to selected configurations. For this reason a residue-based implementation of MRD-CI on a massively parallel architecture offers a good balance between the computational demands and storage requirements.

In the *iteration phase* the full residue tree for all selected configurations must be built, but a single copy of the tree can be distributed across all nodes. All matrix elements associated with a given transition residue can be locally evaluated if all associated orbital pairs are present on a unique node. We note that the residue tree itself (Fig. 3B) is not required at all; only the set of connected orbital pairs is needed. As a result, no lookup operations are required in this step and one can simply loop over the locally available section of the orbital pair segments to evaluate all matrix elements that can be constructed for the present orbital sets. Because each matrix element is uniquely identified by its transition residue, the contributions to the many-body field can be simply collected at the end of this step on a single node to perform the Davidson iteration. This mechanism allows a rapid evaluation of all matrix elements while using the available core memory to its fullest extent.

PARALLEL IMPLEMENTATION

In a truly scalable implementation great care must be taken to divide all work equally across the participating nodes. A remaining nonscalable portion of 1% of the computational effort of a single

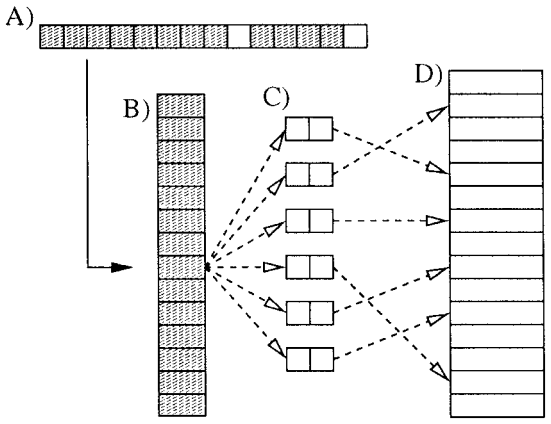

FIGURE 3. Schematic representation of the computation of two-particle matrix-elements in the expansion step using the residue tree. For a given configuration (A) we form all two-particle residues, which are looked up in the residue tree. In the configuration illustrated in (A) each box represents one occupied orbital, the shaded region corresponds to the transition residue, and the two white boxes correspond to the orbital pair. The $(n_e - 2)$ electron residue configuration is looked up in the residue tree (B). Each orbital pair (C) associated with the residue encodes a matrix element with an element of the configuration list (D). The orbital indices of the required integral are encoded in the orbital pairs in (C); the coefficient of the source configuration is looked up directly in (D). Only one lookup operation is required to compute all matrix elements associated with the given transition residue, and only the subset of matrix elements that lead to selected source configurations are constructed.

processor application translates into a 100% overhead if the same task is distributed across 100 nodes. Our massively parallel algorithm for configurations-selecting MRCI is therefore based on a client server model that strictly separates the calculation from the communication steps. The latter were chosen to require only global communication directives of the underlying MPI communication library that can be expected to execute efficiently on most modern parallel architectures.

According to the above outline the overall work can be broken into two distinct phases that require the same order of magnitude of computational effort. Table I summarizes the most important steps of the configurations-selecting CI procedure. In an expansion step we begin with the distribution of the current state vector to all nodes (D1). Each node then builds the restricted residue tree for all the set of configurations it received in step D1. The effort per node involved in this step (R1) is proportional to $n_e^2 N_{conf}/N_k$; it will therefore scale well with the number of nodes N_k. Next (step G1) the residue tree must be distributed to all nodes. Because orbital pairs belonging to the same transition residue have been created on several different nodes, this is a nontrivial operation. Using a hashing mechanism we first assign a unique node to each transition residue and then gather all orbital pairs belonging to that residue on the appropriate node. Then each node builds its unique section of the residue tree where the hashing mechanism ensures the balancing of the computational effort across all nodes. Finally, the information of all the nodes is distributed via an all to all communication across the entire machine. Now (step E) each node can run through a predetermined section of the search space to evaluate the energy contributions and to select the configurations for the variational subspace. Step E dominates the overall computational effort of the configuration selection by a large margin.

The next three steps prepare the variational subspace for the iterations. Because the distribution of the selected configurations on the different nodes can be rather uneven, we first redistribute the configurations among the nodes (D2). Then each node constructs its portion of the full residue tree (R2). These contributions are gathered in analogy to step G1 across all nodes, such that each node has all orbital pairs for its assigned transition residues. In contrast to step G1, however, the entries are not distributed across the machine, instead remaining on the nodes for the local matrix element evaluation. The computational effort in the evaluation step of the matrix elements will be proportional to the expectation value of the square of the number of orbital pairs over the transition residues. The hashing mechanism we used to assign transition residues to nodes, however, ensures only that there are approximately the same number of residues on each node. Figure 4a demonstrates that the computational effort can nevertheless vary quite significantly among the nodes, an effect that worsens with an increasing number of nodes. Such an imbalance in the work distribution leads to the loss of scalability of the algorithm. It is therefore important to redistribute the workload among the nodes to achieve better performance. To this end we gather discretized histograms of the transition residue distribution on the server node, which uses this information to assign approximately even work loads to all nodes. Based on this technique the theoretical deviations in the variation of the work load can be reduced from over 50% to less than 12% of the overall average

TABLE I.
Distinct Computational Steps in Parallel Implementation of Configuration-Selecting MRCI Procedure.

	Phase		Type
I	Initialization	All nodes	Computation
	Expansion & Logic Steps		
D1	Distribution of initial state	All nodes	Broadcast
R1	Build section of restricted residue tree	All nodes	Computation
G1	Gather and distribute residue tree	All nodes	Pairwise
E	Expansion loop	All nodes	Computation
D2	Distribute selected configurations	All nodes	All to all
R2	Build one section of full residue tree	All nodes	Computation
G2	Gather the residue tree	All nodes	All to all
	Iteration Steps		
D3	Distribute new coefficients	Node 0	Broadcast
M	Evaluate matrix elements	All nodes	Computation
G3	Gather many-body field	Node 0	Gather
X	Davidson iteration	Node 0	Iteration

There are three phases associated with the initialization of the program and the expansion and iteration of the state vector. The details of the phases are discussed in the text. The third column of the table details which set of nodes is involved in each step, while the fourth column indicates the type of operation that dominates the step. Almost all the computational effort is concentrated in steps E and M, the next leading contribution arising from the logic steps D2, R2, G2. Only standard high-level communication routines were used to make the program as portable as possible.

computational effort. Figure 4 shows histograms of the distribution of the work load with and without active load balancing to demonstrate this observation.

After the redistribution of the transition residue table, the program can proceed with the iteration steps to converge the variational subspace. The four steps comprising an iteration are executed many times after each expansion loop, but they require no further logic information. Almost all the work is concentrated in step M. We note in passing that in the expansion step only a fraction of all possible integrals are required on the nodes, a fact that will be exploited in future versions of the code.

Benchmark Calculations

In order to demonstrate the scalability of the implementation we conducted benchmark calculations for typical applications of the program. The first example is concerned with the evaluation of the importance of the triple and quadruple excitations for the PESs of the oxygen molecule and its anion. Previous work established that the accurate calculation of the electron affinity of O_2 remains a formidable challenge, even for present day quantum chemical techniques. At the level of a CAS-SCF description the adiabatic electron affinity of the oxygen molecule is predicted with the wrong sign, even in the basis set limit. A careful study[28] concluded that strong differential dynamical correlation effects are most likely entirely responsible for the source of this discrepancy. In MRCI-SD calculations the correct sign for the electron affinity can barely be reached using aug-QZP quality basis sets. A semiquantitative agreement between experiment and theory was reached when the multireference generalization of the Davidson correction[6] was applied to estimate the effect of higher excitations.

Therefore, O_2 is one of the simplest molecules that challenges one of the central paradigms of modern quantum chemical correlation methods that rest on the assumption that the explicit treatment of single and double excitations of a chemically motivated reference set is sufficient to quantitatively account for dynamical correlation effects. This observation, as well as the desire to explicitly test approximations for extensivity corrections to MR-SDCI,[7–10] motivated the development of the present code. Because the CAS + SDTQ Hilbert space of O_2 in an aug-QZP basis has the dimen-

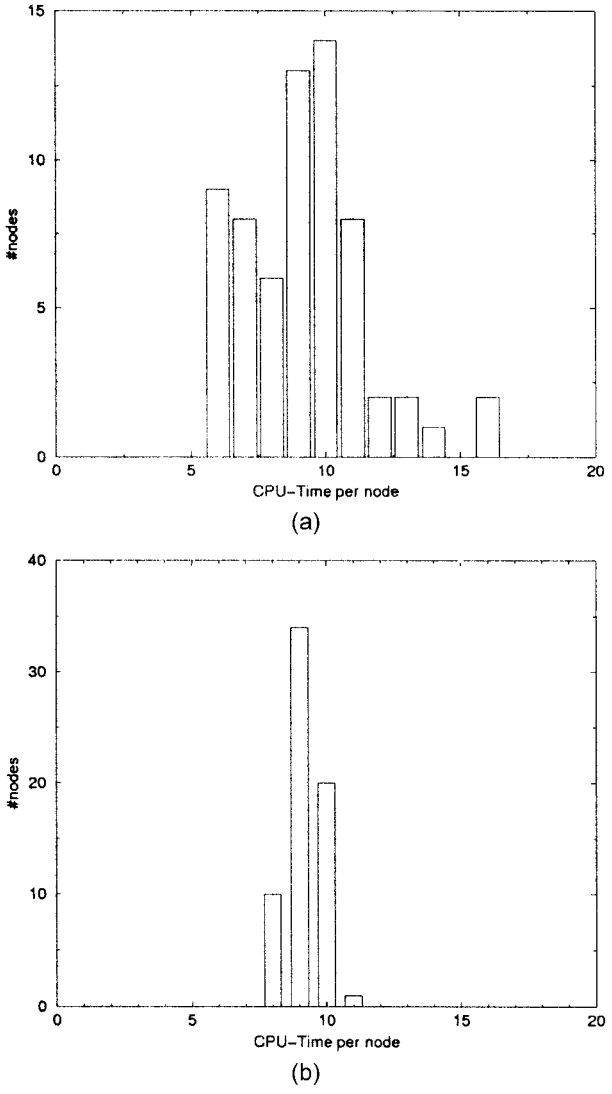

FIGURE 4. Histogram of the CPU time in the iteration step for the O_2 benchmark calculation including triple and quadruple excitations described in the text (a) in the absence of load balance through the exchange of residue entries between nodes and (b) with such load balancing. The reduction in the width of the distribution improves the scalability of the algorithm.

sion 32×10^9, this problem cannot be treated with any of the presently available MR-SDCI or MRD-CI implementations; but it provides a suitable challenge for our parallel implementation. The calculations were performed in an (sp)-augmented cc-pVTZ, cc-pVQZ, and cc-pV5Z[28,29] basis set in D_{2h} symmetry at the experimental geometries. In Hilbert spaces of dimension up to 5×10^9 containing triple and quadruple excitations we selected up to 5×10^6 determinants as a function of the threshold for the coefficients ranging from 10^{-3} to 10^{-6}. The convergence of the treatment of TQ excitations as a function of the selection threshold for an aug-cc-pVTZ basis is demonstrated in Figure 5. The electron affinity of O_2 at the level of SD excitations extrapolates to the wrong sign (+2.8 mH). A quantitatively correct calculation must therefore include an adequate treatment of TQ excitations. The multireference Davidson correction yields better agreement with the experimental result, but it differs quantitatively from the explicit inclusion of TQ excitations.

In the second example, which is more within the traditional applications of configuration-selecting CI, we computed the ground state energy of benzene in a cc-pVDZ basis set using active spaces of six and 12 active orbitals. The latter calculation was motivated by the desire to test the program for very large Hilbert spaces, but the smaller active space is sufficient to adequately describe the chemistry of benzene. The calculation was performed in a C_{6v} geometry in D_{2h} symmetry, resulting in Hilbert spaces of up to 3×10^9 determi-

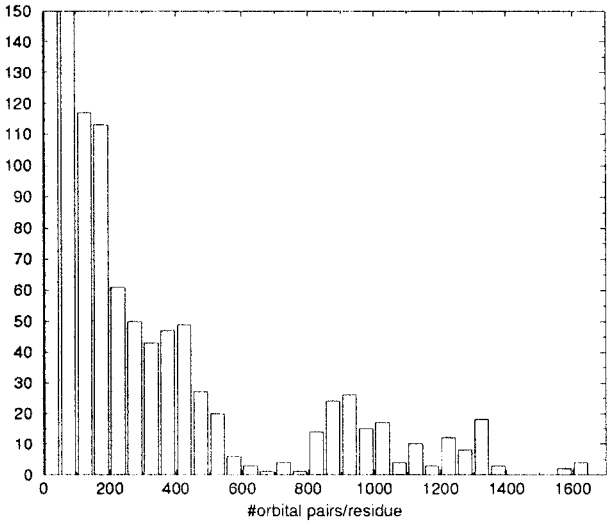

FIGURE 5. Histogram of the distribution of the number of orbital pairs per transition residue in the iteration step for the O_2 benchmark calculation described in the text. The computational effort per transition residue is proportional to the square of the number of orbital pairs. This dependence makes our algorithm efficient, because the construction of the residue tree is much less costly than the evaluation of the matrix elements. A wide distribution, on the other hand, makes it difficult to balance the load in the iteration step. Because the number of transition residues / node decreases with the number of nodes, fluctuations in the computational effort become more difficult to balance for a large number of processors.

nants of which up to 2×10^6 were selected for the variational subspace.

SCALABILITY

The most important consideration in the evaluation of the performance of a parallel program is its scalability with the number of processors used for a given calculation. For scaling purposes we selected a typical run with 10^9 determinants (1.8×10^6 selected) for O_2 and another with 1.3×10^9 determinants (1.6×10^6 selected) for benzene. For these cases we performed benchmark runs on the 256-node IBM-SP2 of the Karlsruhe supercomputer center. We also tested the program on the 256-node and 512-node Cray-TE3s of the supercomputer center (HLRZ) of the Research Center Jülich and used the maximally available number of processors for standard runs (i.e., 256 on the Cray-TE3 and 128 on the IBM-SP2). Unfortunately, the T3E consists of two machines of different physical characteristics: the smaller cluster (128 nodes) permits runs ranging from 16 to 64 nodes; the larger one allows runs requiring 65–256 nodes. The interpretation of the scaling data will have to take this ''break'' into account. The runs on the Cray-T3E with its larger data types but smaller core memory per node forced us to use a somewhat smaller threshold than on the IBM-SP2 for the scaling runs in order to be able to finish the calculation even for a small number of processors. Because all data except integrals and state vectors are distributed across the machines, the size of the maximally treatable Hilbert space grows significantly with the number of nodes.

Figure 6 shows the total computational effort (excluding the time to read the integral file) of the aforementioned scaling runs on the IBM-SP2 as a function of the number of nodes. In these plots the computational effort for all logic-steps sections (D1, R1, G1, D2, R2, G2) are subsumed in one category; the expansion loop (E) and the iteration loop (D3, M, G3, X) constitute the other main components of the program. This division is motivated by the fact that the relative importance of these three main computational steps varies with the type of calculation performed, and a different behavior of these steps will result in changes of the overall performance for different calculations. The number of times the expansion loop is executed, for instance (in the test calculations twice for the SD and once for the TQ segment), will significantly affect the overall performance. Varying the threshold leads to a redistribution of the effort

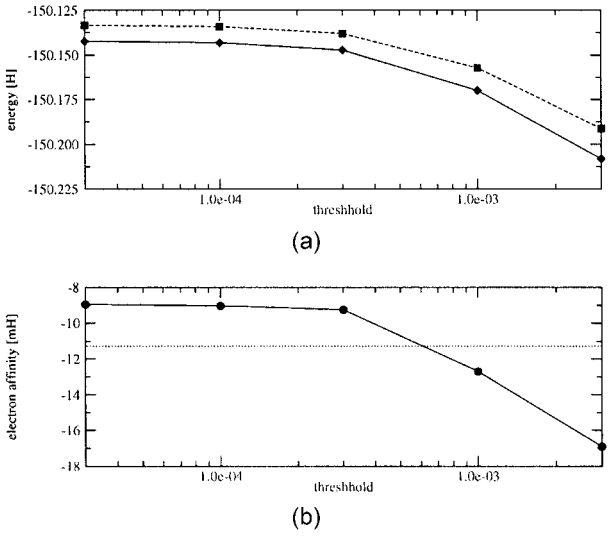

FIGURE 6. (a) Total energy of the(—)oxygen molecule and (- - -) its anion at their experimental geometries including triple and quadruple excitations using an aug-cc-pVTZ basis set as a function of the coefficient selection threshold. (b) Estimates of the electron affinity of the oxygen molecule as a function of the coefficient selection threshold from the calculations in (a). (···) The estimate of the electron affinity from an SD calculation using the multireference generalization of the Davidson correction.

from expansion to matrix elements. Increasing the number of states (only one state per symmetry was computed in the test calculations) will reduce the importance of the logic section, as does an increase in the desired accuracy for the state vector in the iteration steps.

For benzene we found almost perfect scaling from 48 to 128 nodes for the IBM-SP2. The total computational effort in the expansion loop, which dominates the overall computational effort, is constant to within 0.4% when going from the smallest to the largest number of nodes. In contrast, the effort associated with logic and communication grows somewhat with the number of nodes. This is to be expected, because the communication cost grows with the number of nodes and a total of 3.7 and 9.1 GB of data have to be transmitted across the machine for the small and large residue tables, respectively. The overall speedup factor from 64 to 128 nodes is 1.86 (see Table II).

For the benchmark calculation of O_2 a more pronounced increase in the overall computational effort is observed when going from 32 to 128 nodes, in particular with the last doubling from 64 to 128 processors. Again, the computational effort

TABLE II.
Total CPU Times for Benchmark Calculations Described in the Text as Function of Number of Nodes.

	IBM-SP2			
	O_2		C_6H_6	
No. nodes	Time (s / Node)	Loss (%)	Time (s / Node)	Loss (%)
16	8308			
32	4374	5		
48			10122	
64	2480	13	7510	0
96			5160	3
128	1410	14	4012	4
	Cray-T3E			
16	5334			
32	2887	8	319	
64	1539	10	172	8
65	2062		244	
128	1107	6	147	20
256	620	12		

The time for the expansion and convergence of a single state in each calculation is given. The fractional computational loss between two test runs is defined as the ratio of the CPU times per node divided by the perfect speedup factor given by the ratio of the nodes. The loss data in the table always refer to successive entries. The calculation for benzene on the IBM-SP2 employed 12 active orbitals, and that on the Cray-T3E used a realistic active space of six orbitals. Note that the sensible limit for the latter calculations lies around 64 nodes, where less than 5 min are required to converge the calculation.

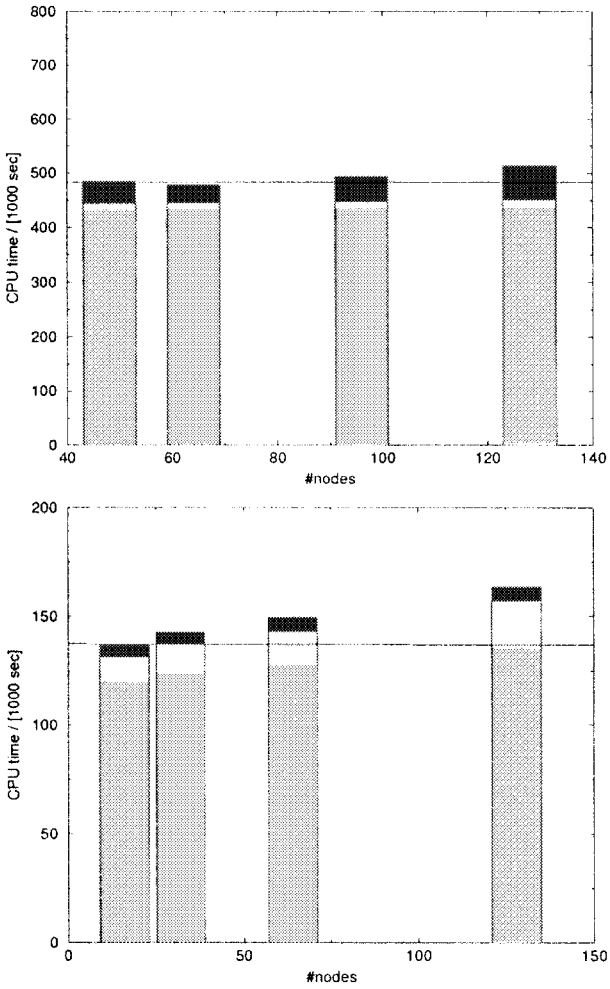

FIGURE 7. Total CPU time for the fully converged calculation of the ground state of the two benchmark calculations described in the text as a function of the number of nodes of the IBM-SP2. A straight line indicates perfect scaling of the computational effort with the number of nodes. The shaded areas in the bars, from top to bottom, indicate the contributions of the matrix element evaluation (steps D3, M, G3, X in Table I), logic (D1, R1, G1, D2, R2, G2), and the expansion loop (step E).

in the expansion loop is relatively stable, increasing only 6.7% (6.3%) when going from 32 to 64 (64 to 128) nodes, respectively. This is the result of the near perfect load balancing in evidence in Figure 7, which shows a histogram of the total CPU time/node for the expansion step for a run on 64 nodes. The presence of TQ excitations significantly complicates the overhead associated with the generation of the residue trees. Given the relatively wide distribution of the computational effort per transition residue (see Fig. 8) that results from the presence of the TQ excitations, it becomes more and more difficult to balance the computational load in the matrix element step. This results in a larger variation in the load (see Fig. 7) among the processors. This decreases the performance of the iteration step, because all processors must wait for the last node to finish. Because large amounts of data are required for the evaluation of the matrix elements, it is difficult to go beyond the present implementation and to dynamically adjust the load while the iteration is in progress.

The data for the test runs is summarized in detail in Table II. For the IBM SP-2 we found near-perfect speedups for benzene. For O_2 the speedup factors associated with the doubling of the nodes were somewhat worse, but they still warrant the use of a large number of nodes to perform the calculation in most circumstances. MRCI calculations require nontrivial communication steps on parallel machines, so that some loss

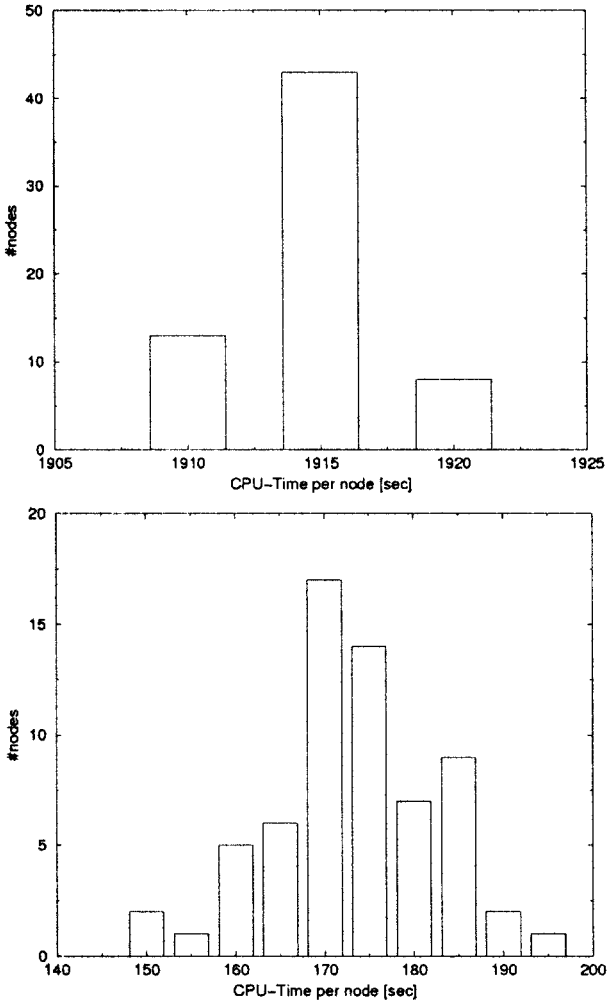

FIGURE 8. Histogram of the CPU time distribution of (a) an expansion step including TQ excitations and (b) an iteration step for the O_2 benchmark calculation described in the text on 64 nodes of the IBM-SP2. The computational effort in the expansion step is almost perfectly distributed; that of the iteration step varies with a standard deviation of approximately 4%, resulting in a loss of computational efficiency as ''fast'' nodes have to wait for the ''slow'' nodes to finish. Without the use of load balancing the width of distribution for the iteration step increases significantly. Because the number of transition residues / node decreases with the number of nodes, fluctuations in the computational effort become more difficult to balance for a large number of processors.

of computational efficiency is unavoidable. On the Cray-T3E we found a similar situation: for benzene we report on calculations with the more realistic six-orbital space. On 128 nodes this calculation requires less than 3 min total turnaround time and the residue table in the iteration step is spread so thinly that it becomes impossible to balance. This explains the somewhat large loss of 20% efficiency in going from 65 to 128 nodes. Note that the data for 64 (65) nodes were obtained on the small (large) cluster of the T3E described above. The time differences are indicative of the relative performance of these two machines.

PERFORMANCE

Having verified the scalability of the method, we now turn to the overall performance of the implementation to give some impression on the total CPU requirement for some well-studied molecules. A great deal of caution is required in the interpretation of this data in a rapidly changing hardware landscape. Modifications in processor speed and architecture, caching capabilities, and memory structure make comparisons between test calculations very difficult. Keeping these shortcomings in mind, however, the overall performance data provides the prospective user with an order of magnitude estimate of the computational requirements. The benchmarks were performed on an IBM-SP2 with 120-MHz RISC2 processors and 512 MB/node and the Cray-TE3 with 300-MHz DEC alpha processors and 128 MB/node. Even though there is almost a factor of 3 between the cycle rates of these machines, the IBM processor has the larger floating point throughput. Neither of these processors is at the high end of present-day computational performance.

Table III summarizes total CPU times for calculations on O_2, NO_2, and benzene for a number of basis sets. Particular attention should be paid to the data for NO_2 in the cc-pVTZ basis that happens to have the same number of basis functions as the basis employed to benchmark the table-direct MRD-CI program and that also refers to the same state and geometry.[21] We adjusted the threshold such that a Hilbert space of almost the same size was selected. Then our total CPU requirement of ca. 5800 s compares favorably with the 24,000 s of TDCI-SAF and the 21,404 s of TDCI-PDD. Similarly, we adjusted the threshold in the benzene calculation to select about the same number of determinants as in the C_6 benchmark for DIESEL-CI[22] where our 2256 s/iteration compare with 2166 s/iteration for DIESEL-CI. Given the different computational architectures employed, these comparisons are inadequate to rate these implementations. Nevertheless, we present them here to demonstrate that our parallel implementation is competitive with modern single-processor implementations of configuration-select-

TABLE III.
Performance Data for Selected MRD-CI SD Calculations.

	Basis	N_{det}	N_{sel}	Nodes	CPU (s)
O_2^-	aug-cc-pVDZ	1×10^5	5.1×10^4	8	24
	aug-cc-pVTZ	1×10^6	3.5×10^5	65	61
	aug-cc-pVQZ	3×10^7	7.5×10^5	65	226
NO_2	cc-pVDZ	3×10^5	9×10^4	8	80
	cc-pVTZ	2×10^6	4×10^5	64	127
	cc-pVQZ	9×10^6	5×10^5	64	1225[a]
C_6H_6	cc-pVDZ	4×10^7	2×10^5	64	172

All calculations were performed on the Cray-T3E of the HLRZ Jülich. The calculation on O_2 was performed in D_{2h} symmetry at the experimental geometry using a (2s2p) complete active space and a coefficient threshold of $\lambda = 10^{-5}$. The calculations on NO_2 were performed on the X $^1A_1$ ground state in C_{2v} symmetry at the experimental geometry using a (1211) complete active space and $\lambda = 3 \times 10^{-5}$. The calculation on C_6H_6 were performed on the X $^1A_1$ ground state in D_{2h} symmetry using a C_{6v} geometry with $R = 2.79$ Å and a (00002121) active space with $\lambda = 3 \times 10^{-5}$. N_{det} designates the size of the Hilbert space, N_{sel} is that of the selected subspace; the next column indicates the number of nodes used, and the last column is the turnaround time till convergence.

[a]The calculation in the cc-pVQZ basis requires twice the number of iterations of the calculation in the cc-pVTZ basis, which is responsible for the disproportionate increase in CPU time.

ing MRCI and that no resources are wasted by employing a parallel machine.

Discussion

Accurate benchmark methods for the treatment of dynamical correlation effects, such as MRCI, have made a significant impact in the development of quantum chemistry. Because their computational effort rises rapidly (as n_e^6) with the number of electrons, only the use of the most powerful computational architectures ensures their continued relevance to the field. Because massively parallel architectures with distributed memory will yield the highest computational throughput in the foreseeable future, it is worthwhile to pursue the use of these machines for quantum chemical benchmark calculations. The development of a scalable implementation of one of the most popular variants of the MRCI method family on such architectures is one important step in this direction. The present implementation allows the treatment of Hilbert spaces and systems that are larger than those that can be treated on traditional architectures, while significantly reducing the turnaround time for more moderate applications. With the ability to routinely treat Hilbert space exceeding 5 billion determinants, many questions that require a delicate balance of dynamical and nondynamical correlation effects (e.g., in transition metal chemistry) become amenable to the MRCI method.

A spin-adapted implementation of the residue-based MRCI algorithm for MR-SDCI[26] and its configuration-selecting variant is presently under way. We report here on the progress of the determinantal program because we do not expect a significant computational advantage in the spin-adapted code, in particular for the configuration-selecting algorithm including TQ excitations. The reason for this expectation lies in the fact that the number of unpaired spins rises rapidly for complex molecules and highly excited configurations. The size of the representation matrices of the symmetric or unitary groups that are required in spin-adapted implementations grows exponentially with the number of unpaired electrons. In nonselecting MR-SDCI the fact that all possible configuration state functions (CSF) are present allows an efficient implementation. In configuration-selecting MRD-CI the sparseness of the state vector leads to a significant computational overhead in the generation of the representation matrices of the line-up permutations that are necessary to evaluate the matrix elements.[22] We note that FCI codes have long ago abandoned CSF-based implementations[30] in favor of a determinantal approach, even though the number of electrons is strongly limited. With larger Hilbert spaces and an increasing electron number a similar trend may appear for configuration-selecting MRD-CI.

One of the intrinsic bottlenecks of our present implementation configuration-selecting MRD-CI is the difficulty of developing efficient integral-driven matrix element evaluation methods in the iteration step, which ultimately limits the size of the basis that can be employed for these calculations. While many interesting calculations will be possible with the existing code, we are presently exploring the possibility of nonlocal integral storage for basis sets exceeding 400 orbitals. Furthermore, we have been able to formulate a scalable integral-driven algorithm of the residue-based matrix element evaluation scheme for the iteration step. Its implementation, which is presently underway in our group, unfortunately breaks the separation of communication and computation steps that are impor-

tant to insure the scalability of the method *a priori*. Such considerations are not important for the expansion step of MRD-CI with SD excitations, because only a small fraction of the full integral table is required there.

In addition, it may be worthwhile to investigate approximations, such as multireference second-order Brillouin–Wigner perturbation theory,[16,31–33] that eliminate the selected variational subspace in MRD-CI altogether. Such approximations rest on the assumption that the individual energy contributions of the selected configurations in MRD-CI are well estimated perturbatively. MRD-CI nevertheless benefits from their explicit treatment because the selected configurations generate a many-body field on the primary configurations that alters the relative importance of the latter. BW-MRPT incorporates this effect in the framework of perturbation theory and eliminates the selected subspace of MRD-CI. In comparison to MRD-CI, BW-MRPT thus discards the coupling among the selected configurations outside the primary space and the present program is an ideal framework to test the importance of these interactions for a wide variety of systems.

Acknowledgments

This work was supported by computational resources at the HLRZ Jülich and the HRZ Karlsruhe. We gratefully acknowledge stimulating discussions with H. Lischka and I. Shavitt in the course of this work.

References

1. Roos, B. O. Chem Phys Lett 1972, 15, 153.
2. Roos, B. O.; Siegbahn, P. E. M. In Methods of Electronic Structure Theory; Schaefer, H. F. III, Ed.; Plenum: New York, 1994; p. 189.
3. Shavitt, I. In Modern Theoretical Chemistry; Schaefer, H. F. III, Ed.; Plenum: New York, 1977.
4. Sheppard, R.; Shavitt, I.; Pitzer, R. M.; Comeau, D. C.; Pepper, M.; Lischka, H.; Szalay, P. G.; Ahlrichs, R.; Brown, F. B.; Zhao, J. Int J Quantum Chem Symp 1988, 22, 149.
5. Langhoff, S. R.; Davidson, E. R. Int J Quantum Chem 1974, 8, 61.
6. Butscher, W.; Shih, S.; Buenker, R. J.; Peyerimhoff, S. D. Chem Phys Lett 1977, 52, 457.
7. Cižek, J. J Chem Phys 1966, 45, 4256.
8. Bartlett, R. J.; Shavitt, I. Chem Phys Lett 1977, 50, 190.
9. Gdanitz, R.; Ahlrichs, R. Chem Phys Lett 1988, 143, 413.
10. Szalay, P.; Bartlett, R. J. J Chem Phys 1995, 103, 3600.
11. Daudey, J. P.; Heully, J.-L.; Malrieu, J. P. J Chem Phys 1993, 99, 1240.
12. Sánchez-Marín, J.; Nebot-Gil, I.; Malrieu, J. P.; Heully, J. L.; Maynau, D. Theor Chim Acta 1997, 95, 215.
13. Buenker, R. J.; Peyerimhoff, S. Theor Chim Acta 1968, 12, 183.
14. Buenker, R. J.; Peyerimhoff, S. D. Theor Chim Acta 1974, 35, 33.
15. Buenker, R. J.; Peyerimhoff, S. D. Theor Chim Acta 1975, 39, 217.
16. Gershgorn, Z.; Shavitt, I. Int J Quantum Chem 1968, 2, 751.
17. Huron, B.; Malrieu, J. P.; Rancurel, P. J Chem Phys 1973, 58, 5745.
18. Buenker, R. J.; Peyerimhoff, S. D. New Horizons in Quantum Chemistry; Reidel: Dordrecht, 1983.
19. Whitten, J. L.; Hackmeyer, M. J Chem Phys 1969, 51, 5548.
20. Harrison, R. J. J Chem Phys 1991, 94, 5021.
21. Krebs, S.; Buenker, R. J. J Chem Phys 1995, 103, 5613.
22. Hanrath, M.; Engels, B. Chemical Physics 1997, 225, 197.
23. Schuler, M.; Kovar, T.; Lischka, H.; Sheppard, R.; Harrison, R. J. Theor Chim Acta 1993, 84, 489.
24. Lischka, H.; Dachsel, F.; Shepard, R.; Harrison, R. J. In High-Performance Computing and Networking. International Conference and Exhibition Proceedings. Vol. 1: Applications; Gentzsch, W.; Harms, U., Eds.; Springer: Berlin, 1994; p. 203.
25. Dachsel, H.; Lischka, H.; Shepard, R.; Nieplocha, J.; Harrison, R. J. J Comput Chem 1997, 18, 430.
26. Stephan, F.; Wenzel, W. J Chem Phys 1998, 108, 1015.
27. Steiner, M. M.; Wenzel, W.; Wilkins, J. W.; Wilson, K. G. Chem Phys Lett 1994, 231, 263.
28. Gonzáles–Luque, R.; Merchán, M.; Fülscher, M. P.; Roos, B. O. Chem Phys Lett 1993, 204, 323.
29. Peterson, K. A.; Kendall, R. A.; Dunning, T. H. J Chem Phys 1993, 99, 1930.
30. Olsen, J.; Jorgensen, P.; Simons, J. Chem Phys Lett 1990, 169, 463.
31. Wenzel, W.; Wilson, K. G. Phys Rev Lett 1992, 68, 800.
32. Wenzel, W.; Steiner, M. M. J Chem Phys 1998, 108, 4714.
33. Wenzel, W.; Int J Quantum Chem to appear.